Predictive Modeling

Week 6

ACSC412/512

Fall 2017

Richard Xu

What were covered last time

1. Tree model (CART)
2. Clustering – K-means
3. Clustering – hierarchical; insurance application
4. More on project

What we are going to discuss today

1. Homework 4
2. Tree-based model summary
3. Random forests
4. Support Vector Machine
5. Neural network
6. Project Q&A
7. Review & Exam II

Homework4

Problem1

1. glm(LapseR ~ …, family=binomial, weights=exposureN,…)

   glm(cbind(LapseN, ExposureN-LapseN)~…,famile=binomial,…)

   glm(LapseN ~ offset(log(ExposureN))+…,famile=possion,…)
2. Same problem for target, offset, and exposure
3. Same target variable for comparison (RiskClass?), same model
3. AIC/deviance - not comparable for dif distributions
4. PremiumJump – numeric or categorical?
5. Cross-term???

Problem2

1. Simple logistic mode

glm(DamageProb ~ …, family=binomial,…)

glm(cbind(DamageIndex, 20-DamageIndex)~…,famile=binomial,…)

2. DamageIndex -> numeric, not categorical

Prediction at T=31 -> 96%

Summary (CART)

Advantages

- No distribution assumptions on the variables
- Both classification & regression
- Excellent interpretability of tree structure
- Not significantly impacted by outliers in data
- Missing values handled at algorithm level

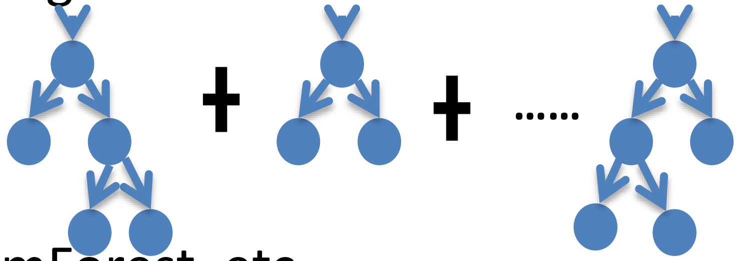
Disadvantages

- CART binary tree, may lead to instability; subject to outlier
- Splits aligned with axes of feature space, may be suboptimal; very low efficiency in linear relationship

CART good, but not good enough…
For example, look at https://www.kaggle.com/competitions - the Home of Data Science

Extensions to Tree Ensembles

- Averaging method - build several estimators independently & average their predictions; the combined estimator is usually better than any of the single base estimator as its variance is reduced.
  - Example – bagging; random forests
- Boosting method- base estimators are built sequentially with reweights for hard to classify objects; combine several weak models to produce a powerful ensemble
  - Example – AdaBoost, Gradient Boosting
- Packages in R
  - All available in R
  - AdaBoost, ipred, mboot, gmb, randomForest, etc.



Bagging - Bootstrap Aggregation

Bootstrap

A sample of size m from the training data is

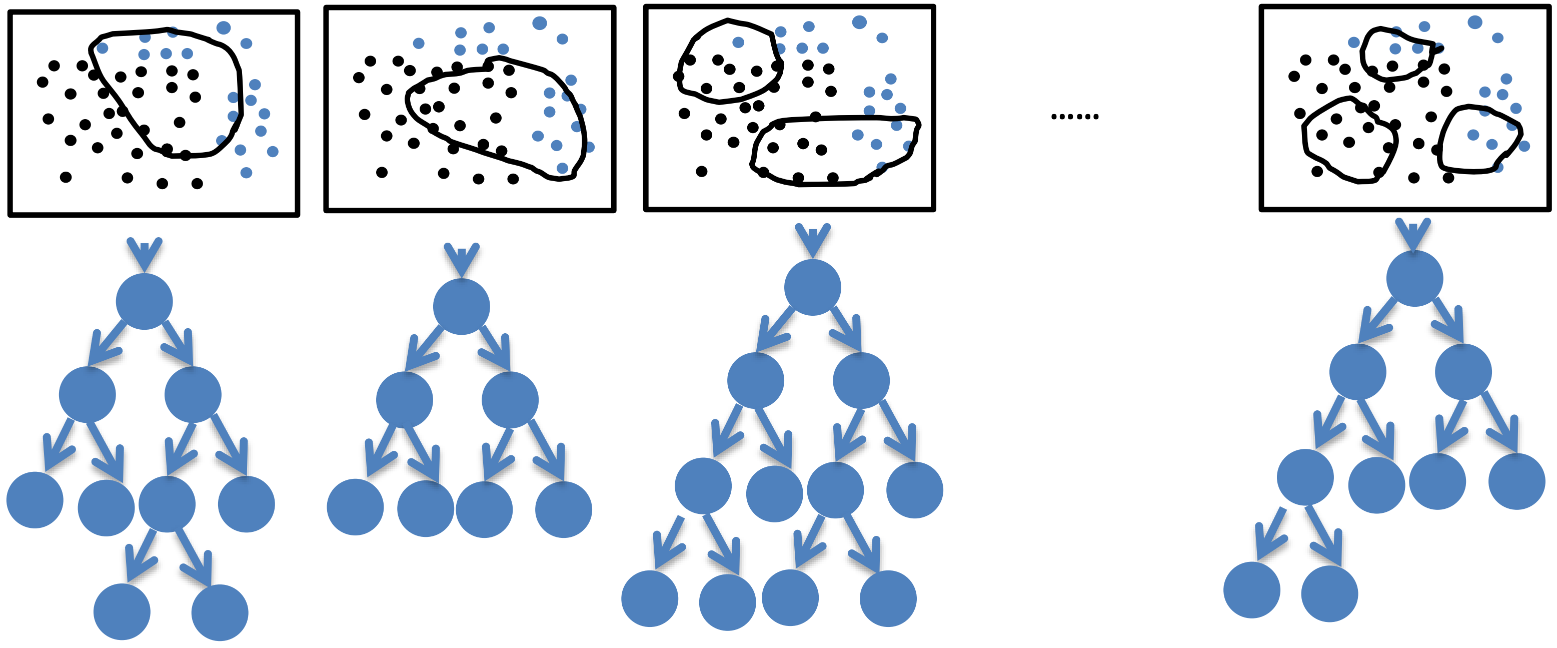
$$(x_i^*, y_i^*), i = 1,2,\ldots,m$$

where each (x_i^*, y_i^*) are drawn uniformly & randomly from $(x_i \ , y_i \ ), i = 1,2,\ldots,n$, with *replacement*

- Equivalent to exactly to m independent draws original data; approximates dataset if we could sample more data points
- For m = n, sample an entirely new training set
- Not all training points are represented in a bootstrap sample, & some are represented more than once
- If m = n & large, ~36.8% of points are left out ($(1 - 1/n)^n \rightarrow e^{-1}$)

Bagging

A sub-sample of the dataset is used to grow a tree. A series of trees could be grown by sub-sampling with replacement.



Bagging - Bootstrap Aggregation

- For i = 1 .. M
  - Draw $n \geq n^*$ samples from *D* with replacement
  - Build classifier C_i
- Final classifier is a vote of C_1 .. C_M or average

- Increases classifier stability/reduces variance
- Decrease the misclassication rate (evaluated on a large independent(?) test sets)
- bagging a good classier can improve accuracy, but bagging a bad one can seriously degrade accuracy
- Computational complexity

Features of Bagging

- It is OK to have individual tree over-fitting
- Individual decision trees are grown deep (prune is not needed)
- Over-fitting can be controlled by the number of trees grown i.e. grow many trees until no improvement is shown
- May take long to grow a series of trees
- Not as popular as random forest and gradient boosting

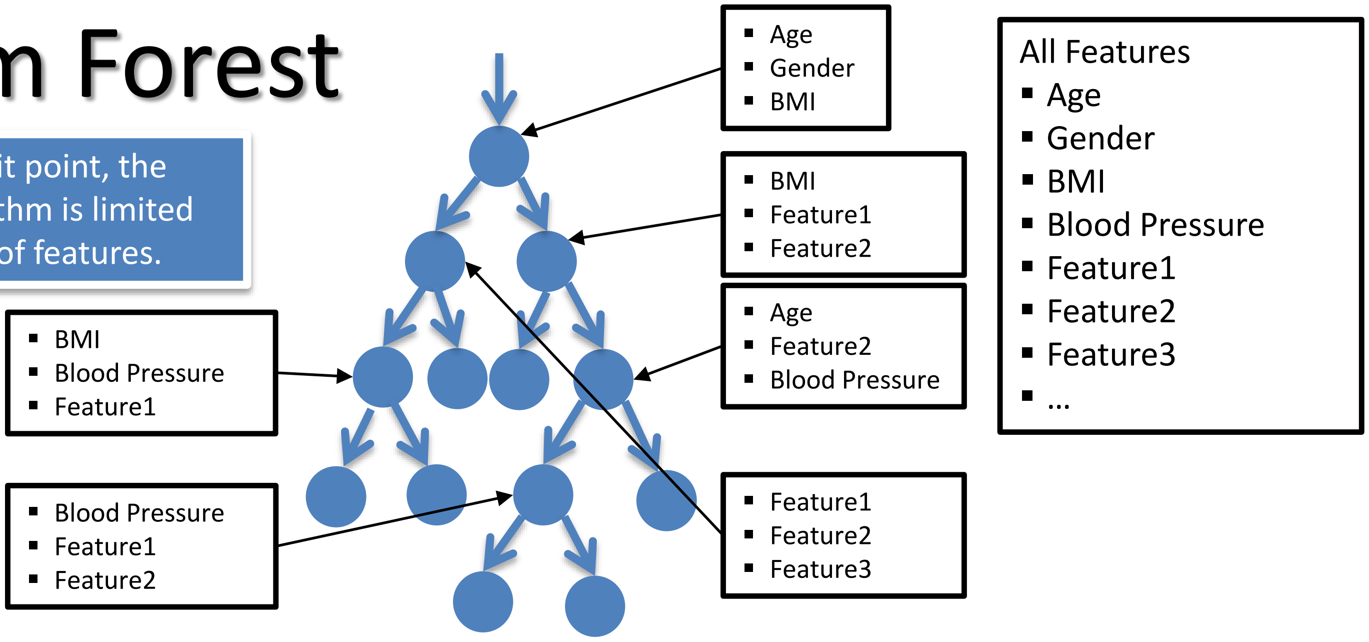
Random Forests

Use random subsets of the predictor variables to grow the trees

- Introduced by L. Breiman in 2001
- Ensemble methods, Dietterich (1999) and (2000)
- Popular & very efficient algorithm of statistical learning, based on model aggregation ideas, for both classification and Regression problems
- De-correlated trees, reduce the variance of final results
- Optimum range of values is often quite wide: number of trees, number of variables

Random Forest

When selecting a split point, the random forest algorithm is limited to a random sample of features.



- A combination of "bagging" and feature selection
- It is OK to have individual trees over-fitting
- Individual decision trees are grown deep (prune is not needed)
- Over-fitting can be controlled by the number of trees grown i.e. grow many trees until no improvement is shown
- May take long to grow a series of trees
- Reducing correlations between trees improves model accuracy

Pros

- Over-fitting is not a problem; no need for pruning trees
- Accuracy & variable importance generated automatically
- Not very sensitive to outliers in training data
- Easy to set parameters

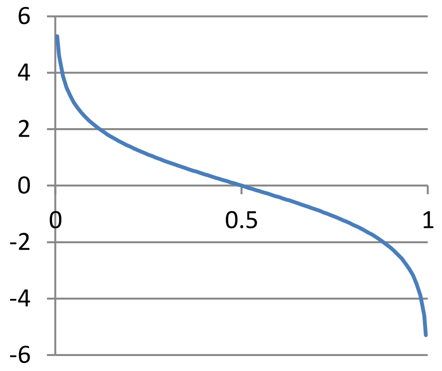
Cons

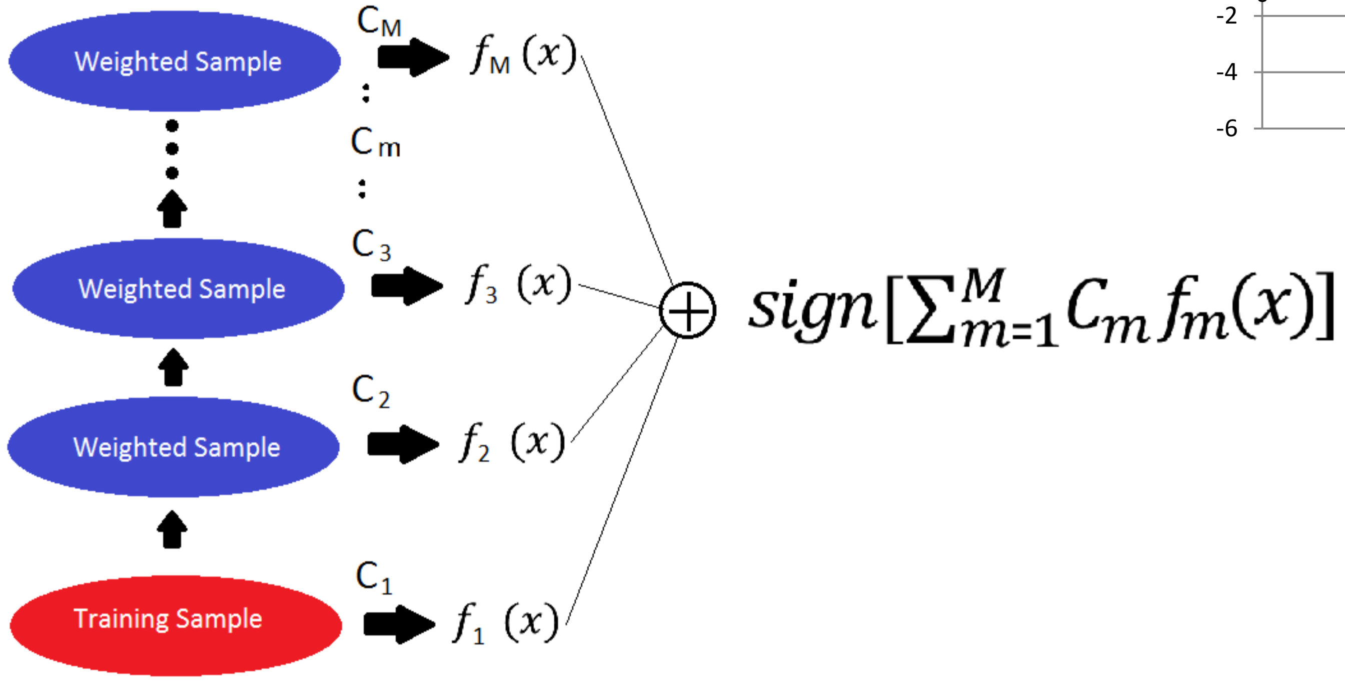
- Regression can't predict beyond range in the training data
- In regression extreme values are often not predicted accurately – underestimate highs & overestimate lows

(example)

AdaBoosting (Adaptive Boosting)

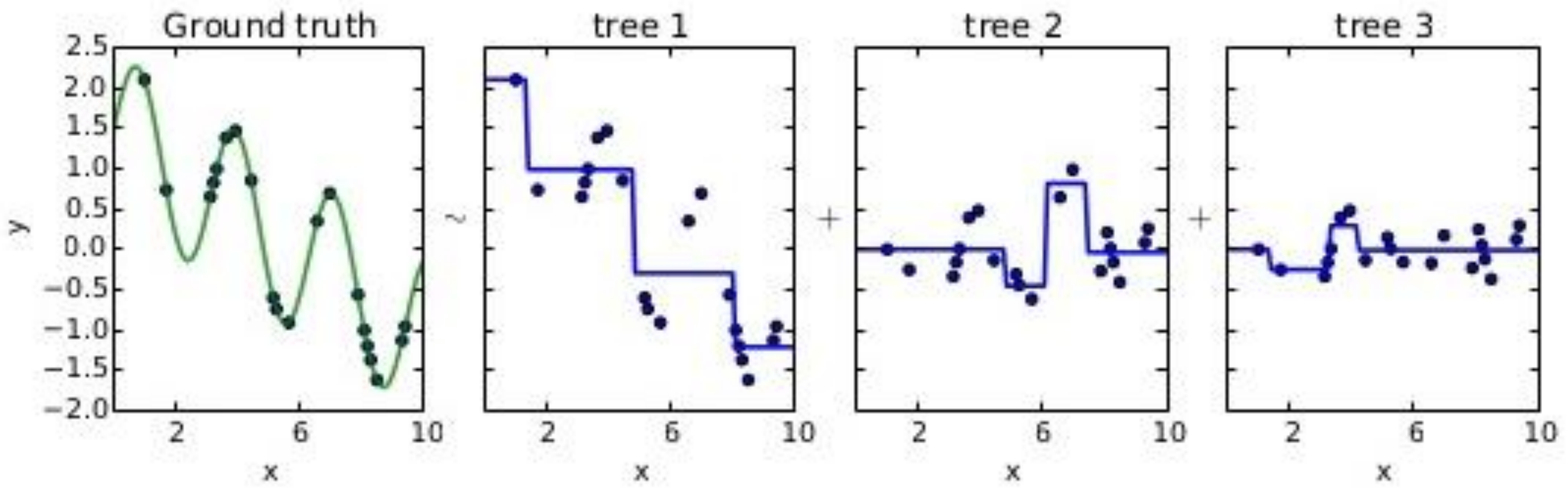
- Initialize weights to be equal $w_i = 1/n$
- For m = 1 to iteration M (tree number)
  - Fit classifier $f_m(x)$ to the weighted data (w_i)
  - Compute the (weighted) misclassification rate $err_m = E[I_{y \neq f_m(x)}]$
  - Let the classifier weight $C_m = \log((1 - err_m)/err_m)$
  - Recalculate weights $w_i = w_i * \exp(C_m * I_{y \neq f_m(x)})$
- Majority vote classification: $sign[\sum_{m=1}^{M} C_m f_m(x)]$





Gradient Boosting

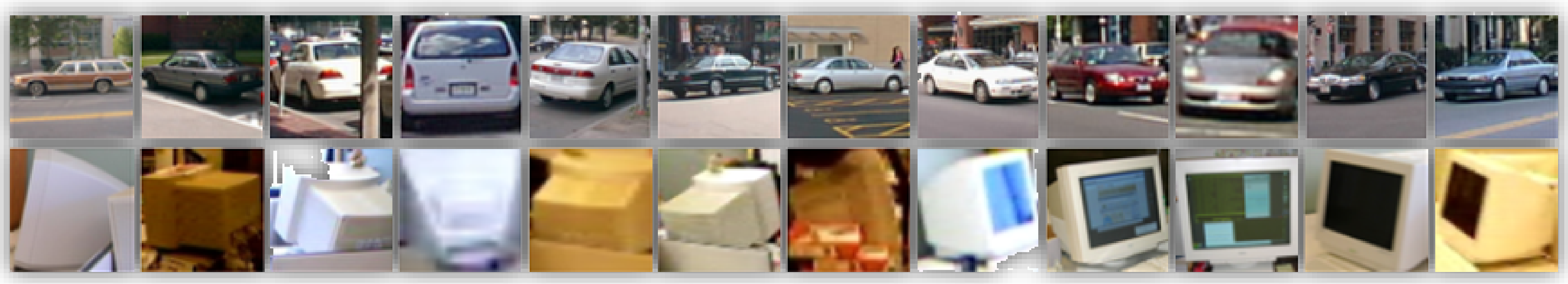
compute a sequence of simple trees, where each successive tree is built to predict residuals of the preceding tree



In Gradient Boosting Algorithm, a series of trees are grown, with each tree fitting the residuals from its precedent trees.

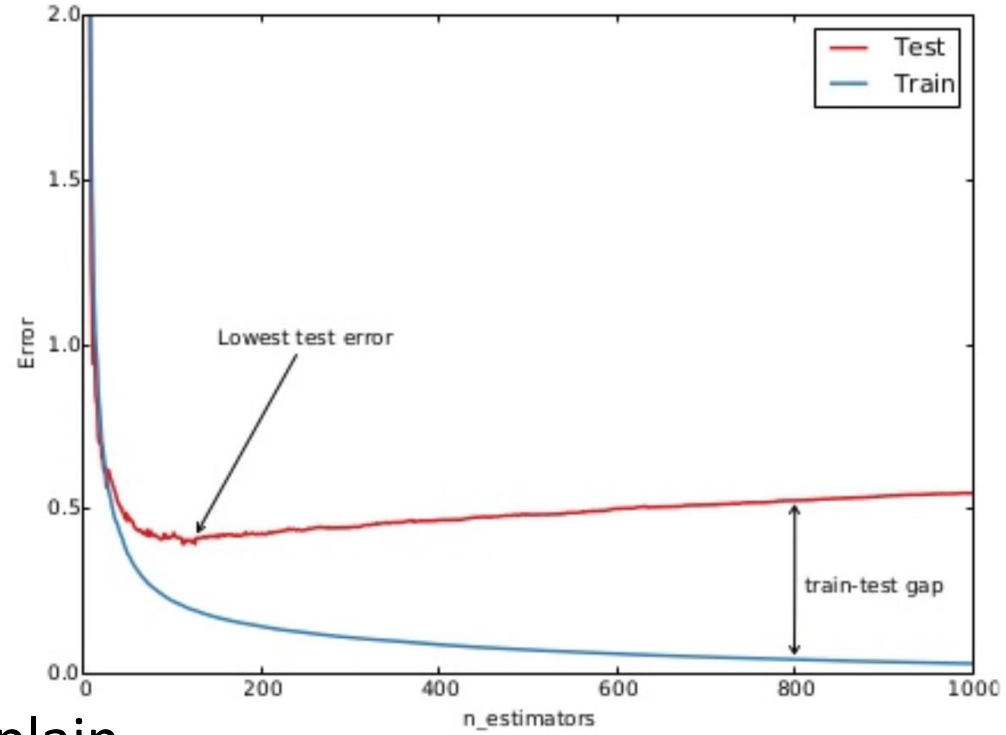
Gradient Boosting

- GBM - each tree is grown to correct the residuals of precedents
- The trees does not need to be a strong estimator by itself
- Predictive power; Robustness to outliers
- GBM could gain accuracy by adding a regularization term to the objective function (Extreme Gradient Boosting Trees)
- Also - Stochastic Gradient Boosting
- GBM - an important component for winning algorithms in Kaggle
- Scalability - due to the sequential nature of boosting it can hardly be parallelized



Modeling procedure

Data split by training/test/validation



Advantage

- (all CART advantages)
- Detect weak pattern
- Flexible non-parametric Classification & Regression

Disadvantage

- Interpretability - hard to explain (partially alleviated by variable importance)
- Require careful turning; more efforts to build (but fast to predict)
- Dangerous to extrapolate

Considerations for Using Tree Algorithms

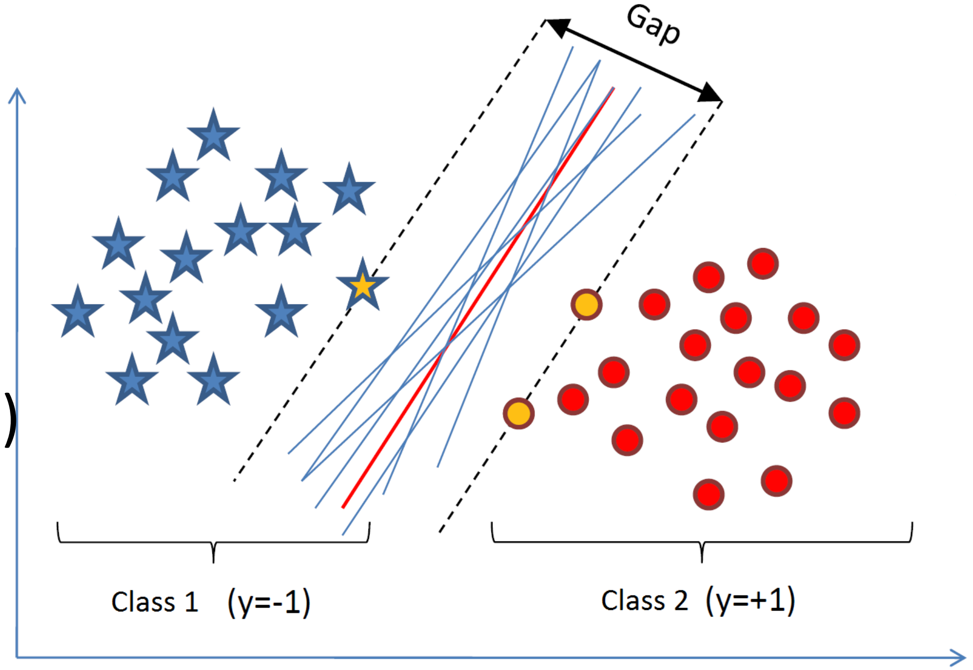
- Random forest and GBM are both important players in machine learning world
- It is critical to pick an algorithm that fits to the inherent structure of a dataset
- Use a simple model if possible
- Use ensemble (complex) model only if necessary
- Control model complexity to avoid over-fitting (validation process is your best friend)
- Build intuition by analyzing model: variable importance, marginal analysis etc.

Support Vector Machine (SVM)

- SVM is related to statistical learning (machine learning)
- SVM was first introduced in 1992
- SVM becomes popular because of its success in handwritten digit recognition (better than NN)
  - Theoretical
    - Robust to very large number of variables but small samples
    - Learn both simple & very complex classification models
    - Employ sophisticated math principles to avoid over-fitting
  - Empirical
    - Superior results
- SVM is now regarded as an important example of "kernel methods", one of the key area in machine learning

Why It is called SVM

Given data $(\overrightarrow{x_i}, y_i)$
$\overrightarrow{x_i} \in R^n$ and
$y_i \in \{-1,+1\}$

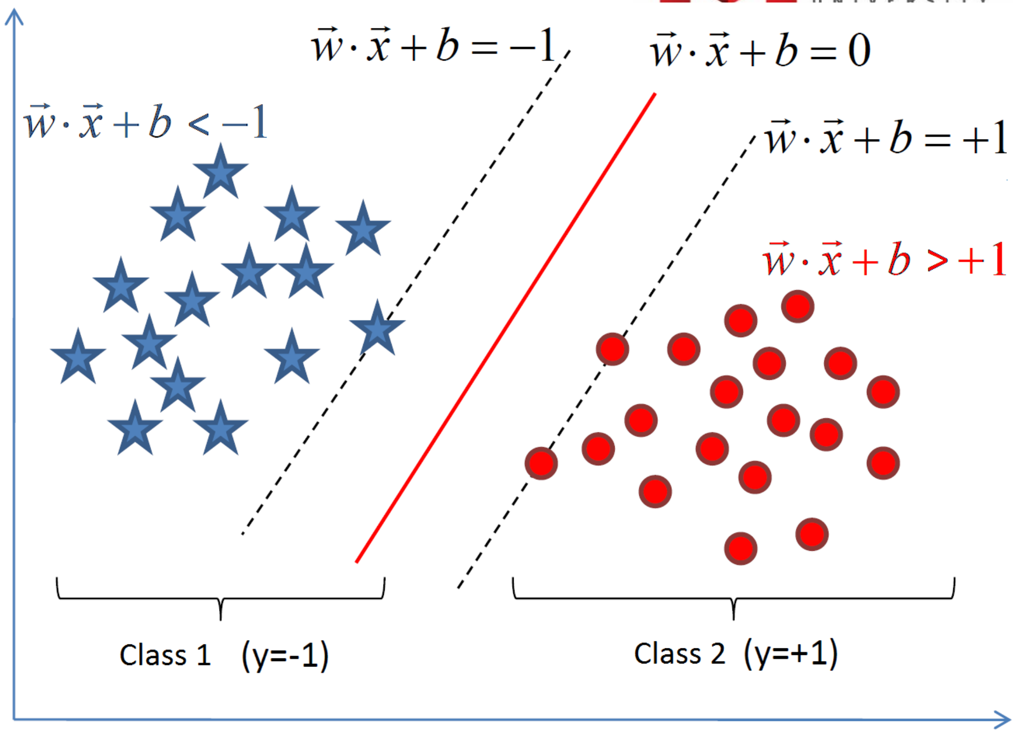


- Find a classifier (hyperplane) to separate 2 classes
- There is a infinite number of such hyperplanes
- SVM tries to find the hyperplane that maximizes the gap between data points on the boundaries (so-called "support vectors")
- If the points on the boundaries are not *informative* (e.g. noise), SVMs will not perform well

Distance between parallel hyperplanes:
$\vec{w} \cdot \vec{x} + b = -1$ and
$\vec{w} \cdot \vec{x} + b = +1$
is

$$D = |b_1 - b_2| / ||\vec{w}|| \\ = 2/||\vec{w}||$$

To maximize gap, it is equivalent to minimize $\frac{1}{2}||\vec{w}||^2$



$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}\cdot\vec{x}+b<-1$
$\vec{w}\cdot\vec{x}+b=+1$
$\vec{w}\cdot\vec{x}+b>+1$
Class 1 (y=-1)
Class 2 (y=+1)

(1/2 is convenient for taking derivative)

We need to impose constraints for correct classification:

$$\vec{w} \cdot \vec{x_i} + b \leq -1 \text{ if } y_i = -1$$
$$\vec{w} \cdot \vec{x_i} + b \geq +1 \text{ if } y_i = +1$$

Or, in concise way $y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1$

Then, classifier is $f(\vec{x}) = sign(\vec{w} \cdot \vec{x_i} + b)$

Primal form

Minimize $\frac{1}{2}||\vec{w}||^2$, subject to $y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1$

A quadratic programming (QP) optimization problem with *n* variables ($w_i, i = 1, \ldots, n$), where *n* is # of features in the dataset.
Solution: numeric analysis

Dual form

If $\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x_i}$, it can be recast as dual form

Maximize $\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j}$

subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$

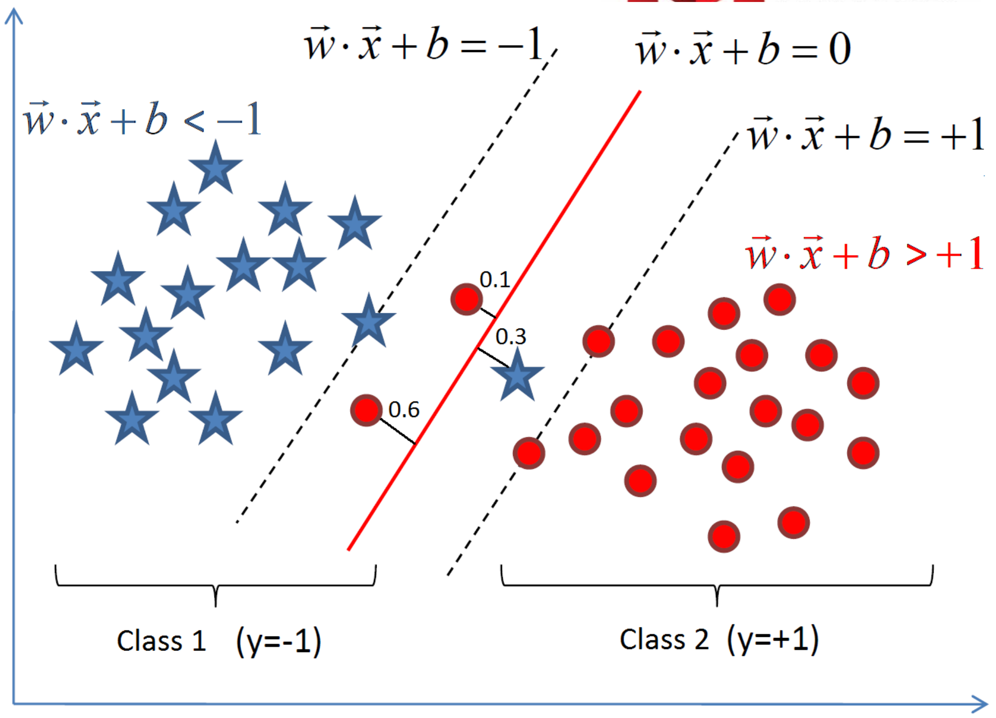
Classifier is $\mathrm{f}(\vec{x}) = sign((\sum_{i=1}^{N} \alpha_i y_i \vec{x_i}) \cdot \vec{x_i} + b)$

No need to access original data, just only data dot products
Number of free parameters is bounded by number of support vectors; not by number of variables (beneficial for high-dimensional problems)

“Soft-margin” linear SVM

What if the data is not linearly separable? e.g. outliers or noisy measurements or non-linear

Assign a “slack variable” to each instance $\xi_i \geq 0$, distance from the separating hyperplane if an instance is misclassified; 0 otherwise.

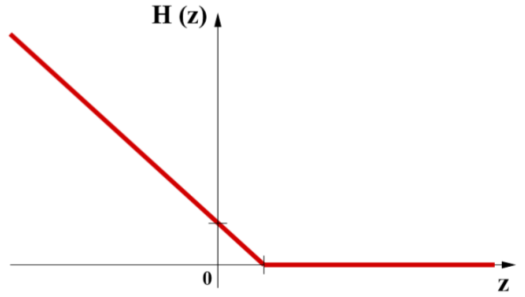


$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}\cdot\vec{x}+b<-1$
$\vec{w}\cdot\vec{x}+b=+1$
$\vec{w}\cdot\vec{x}+b>+1$
0.1
0.3
0.6
Class 1 (y=-1)
Class 2 (y=+1)

Minimize $\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{N}\xi_i$

subject to $y_i(\vec{w}\cdot\vec{x_i}+b) \geq 1-\xi_i$

(also dual form)

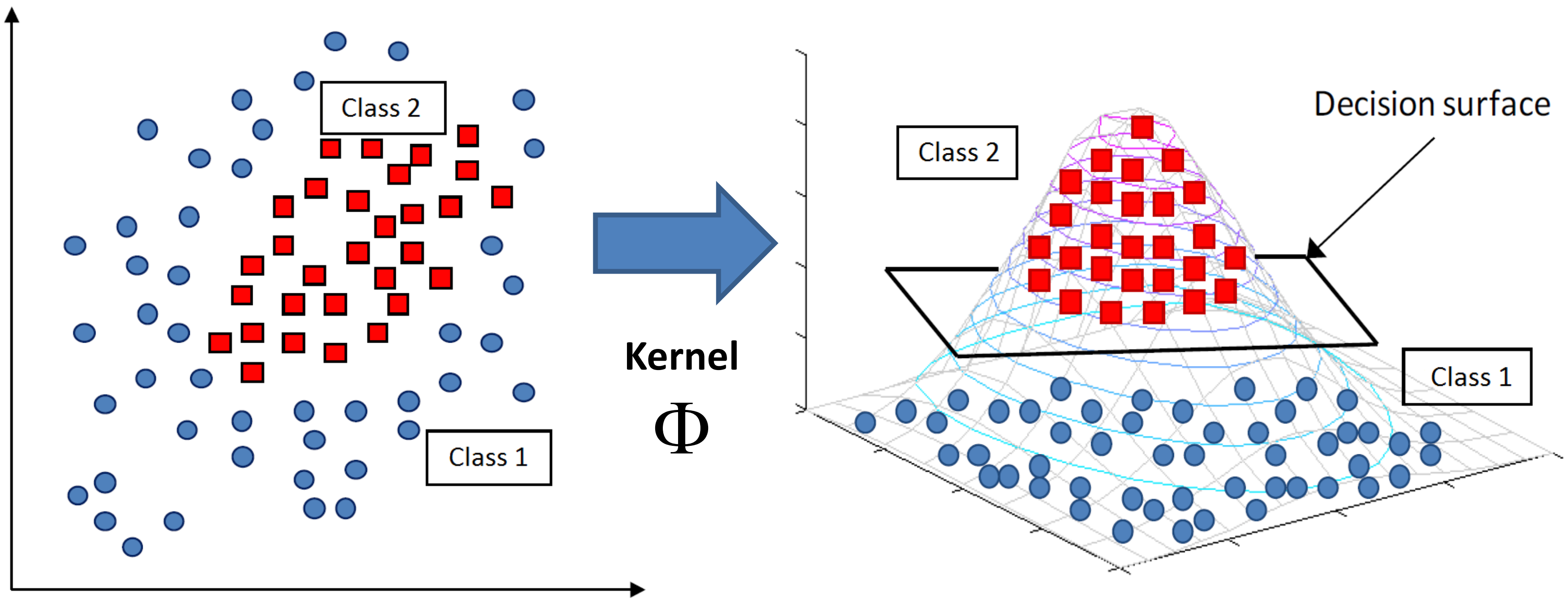


C- cost of misclassification

C large $\rightarrow$ hard-margin SVM

C small $\rightarrow$ allow misclassifications at the expense of small norm of *w*

Not linearly separable data – kernel



Data is not linearly separable in input space, but after transformation, it is linearly separable in the new feature space

Kernel

If $\vec{x_i} \rightarrow \Phi(\vec{x_i})$, dual form can be rewritten as

$$\text{Maximize } \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i\alpha_j y_i y_j \boldsymbol{\Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})}$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0$$

Classifier is $\mathrm{f}(\vec{x}) = sign((\sum_{j=1}^{N} \alpha_j y_j \boldsymbol{\Phi(\vec{x_j})}) \cdot \boldsymbol{\Phi(\vec{x_i})} + b)$

So, if we define $K(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{x_j}}) = \boldsymbol{\Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})}$, we do not need to know Φ explicitly, but to define the function $K(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{x_j}})$

A kernel is a dot product in feature space; example

Linear kernel	$K(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{x_j}}) = \vec{\boldsymbol{x_i}} \cdot \vec{\boldsymbol{x_j}}$
Gaussian (RBF) kernel	$K(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{x_j}}) = \boldsymbol{exp}(-\boldsymbol{\gamma}\|\|\vec{\boldsymbol{x_i}} - \vec{\boldsymbol{x_j}}\|\|^2)$
Exponential kernel	$K(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{x_j}}) = \boldsymbol{exp}(-\boldsymbol{\gamma}\|\|\vec{\boldsymbol{x_i}} - \vec{\boldsymbol{x_j}}\|\|)$
Polynomial kernel	$K(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{x_j}}) = (\boldsymbol{p} + \vec{\boldsymbol{x_i}} \cdot \vec{\boldsymbol{x_j}})^q$

Other include Hybrid kernel, Sigmoidal

(Gaussian - radial basis function (RBF) kernel)

Understand Kernel

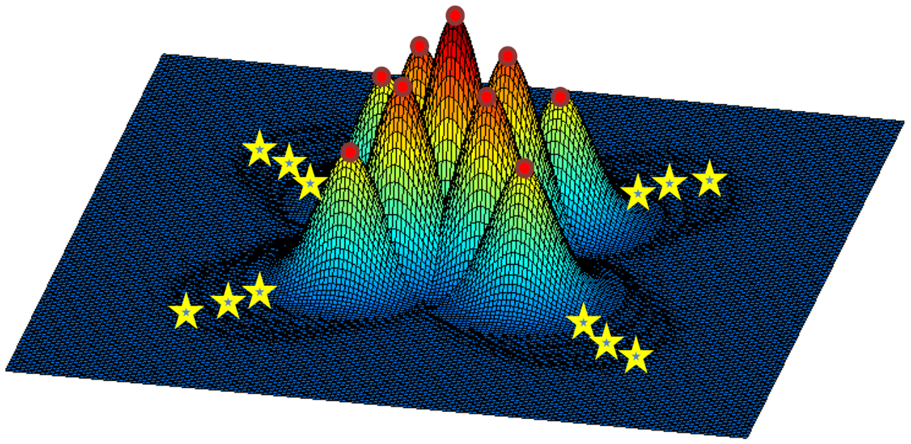
Take Gaussian kernel as example

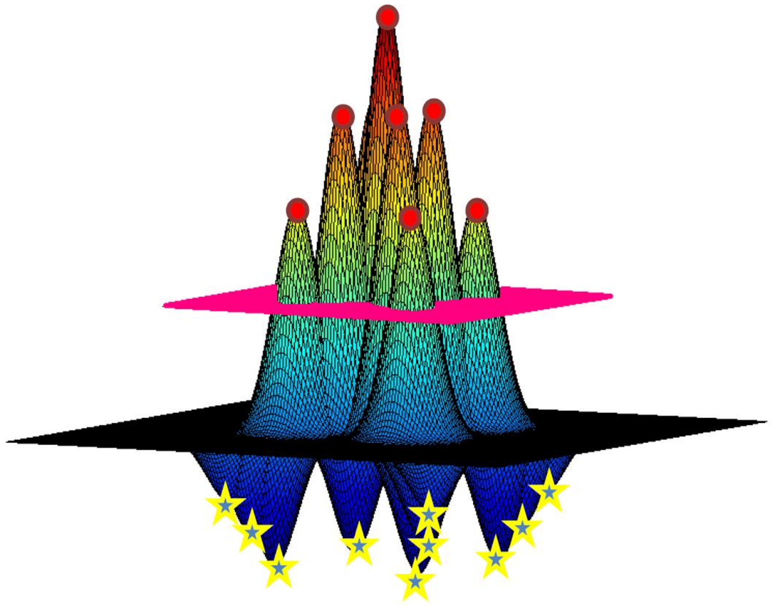
$$K(\overrightarrow{\boldsymbol{x_i}}, \overrightarrow{\boldsymbol{x_j}}) = \boldsymbol{exp}(-\gamma||\overrightarrow{\boldsymbol{x_i}} - \overrightarrow{\boldsymbol{x_j}}||^2)$$

Geometrically, this is a “bump” or “cavity” centered at the training data point

The resulting mapping function is a combination of bumps and cavities

Linear hyperplane can separates two classes in new feature space



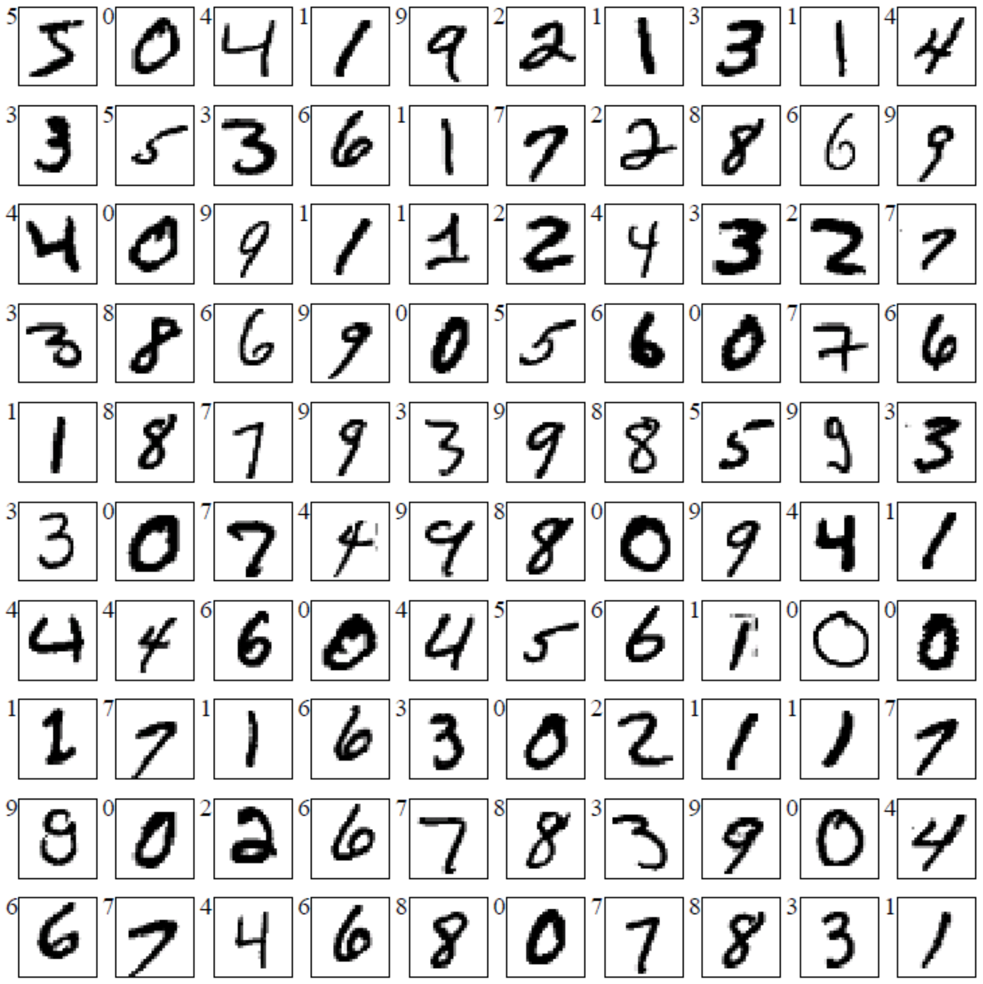


MNIST hand-writing recognition

60,000 training examples, 10,000 test examples, 28x28.

Linear SVM has about 8.5% test error

Polynomial SVM has around 1% test error; RBF ~1.4%



- For most statistic model, need ≥ 5 samples for each parameter to be estimated

- SVMs do not have such requirement & often require much less sample than number of variables, even when a high-degree polynomial kernel is used.

Empirically achieve excellent results in high-dimensional data with very limited samples
Internal capacity control to avoid overfitting
Both simple linear & complex nonlinear functions by “kernel trick”
Robust to outliers and noise (“slack variables”)
Convex QP optimization problem (global minimum & solved efficiently)

Solution is defined only by a small subset of training points ("support vectors")
Number of free parameters is bounded by the number of support vectors and not by the number of variables
Do not require direct access to data, work only with dot-products of data-points.

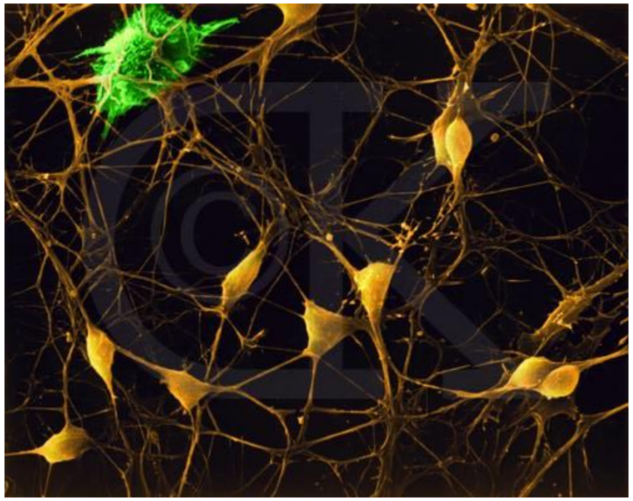
Measures of uncertainty of parameters are not currently well-developed
Interpretation is less straightforward than classical statistics
Lack of parametric statistical significance tests

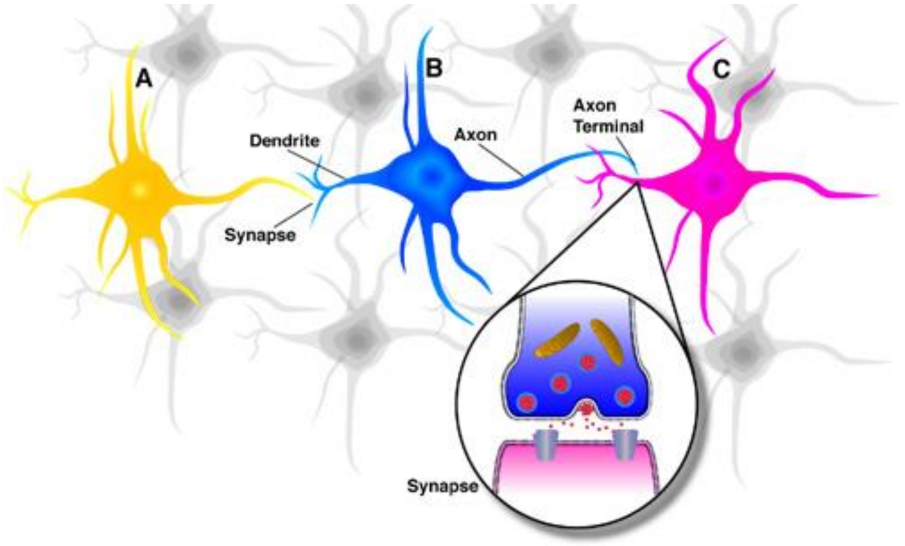
(Example)

Neural Networks
or
Artificial Neural Networks (ANN)

Introduction

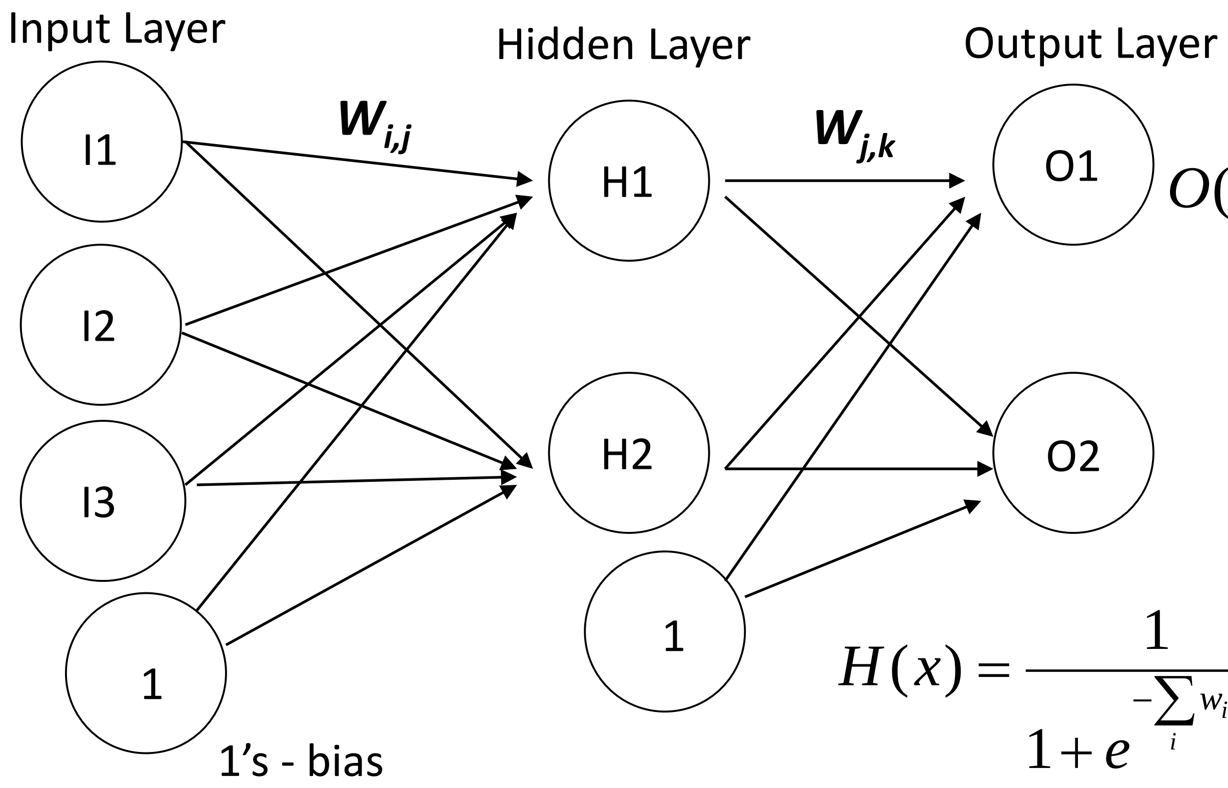
- History back to the 50's but became popular in 80's
  - A General Framework for Parallel Distributed Processing : Explorations in the Microstructure of Cognition
- Peaked in the 90's, until today:
  - Hundreds of variants
  - Less a model of the actual brain than a useful tool
- Numerous applications
  - Handwriting, face, speech recognition
  - Vehicles that drive themselves
  - Models of reading, sentence production, dreaming
- Biological inspiration





Backpropagation Networks

- Attributed to Rumelhart & McClelland in late 70's
- To bypass the linear classification problem, we can construct *multilayer* networks. Typically we have *fully connected*, *feedforward* networks.



$$O(x) = \frac{1}{1 + e^{-\sum_j w_{j,x} H_j}}$$

$$H(x) = \frac{1}{1 + e^{-\sum_i w_{i,x} I_i}}$$

Backpropagation Networks

Procedure

- Randomly assign weights (between 0-1 or -1,1) to each connection
- Present inputs from training data, propagate to outputs
- Compute outputs of model & real data, adjust weights according to the delta rule, back-propagate the errors; weights will be adjusted closer so that the network learns to give the desired output.
- Repeat; stop when no errors, or enough iterations completed

Backpropagation Networks

Modifying Weights

$$\Delta w_k = cI_k\left(T_j - O_j\right)f'\left(ActivationFunction\right); \quad f = \left(\frac{1}{1+e^{-sum}}\right)$$

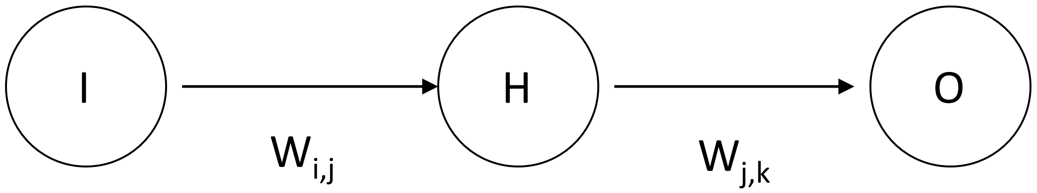
$$\Delta w_k = cI_k\left(T_j - O_j\right)\left(f(sum)(1-f(sum)\right)$$

For the output units, this is:

$$\Delta w_{j,k} = cH_j(T_k - O_k)O_k(1-O_k)$$

For the Hidden units (skipping some math), this is:

$$\Delta w_{i,j} = cH_j(1-H_j)I_i\sum_k(T_k - O_k)O_k(1-O_k)w_{j,k}$$



Backpropagation Networks

Very powerful - can learn any function, given enough hidden units, but that also mean enough data

Have the same problems of Generalization vs. Memorization; w/ many units, it tends to memorizing instead of generalizing; may need to “prune” the neural network

Extensive experience needed, e.g. parameters

Networks require extensive training; possibly extremely slow to train; may also fall into local minimal

Inherently parallel algorithm, ideal for multiprocessor hardware

A very powerful algorithm that has seen widespread successful deployment

MARYVILLE UNIVERSITY

Project

Question?

Some questions to consider

Build Development

- Model & distribution
- Which variables can be used; which can not (think about how you would apply the model, e.g. smoking)
- Grouping of levels for categorical variable (level?)
- Transformation of numeric variables (logarithm & high order polynomial?)
- Simple model or complicated model, why (think of data size & over-fitting) & when
- How to demo predictive power?

Some questions to consider

Model Interpretation

- What we learn from the model; business insights; if it makes sense

Implementation

- Use probability or risk index?
- What is required to implement
- How to mitigate anti-selection (think of model accuracy & associated risks)

MARYVILLE UNIVERSITY

Review

PM general

- What & why; data sources; insurance applications
- PM Terminology & common algorithms
  - Supervised vs. Unsupervised Learning
  - Classification vs. Regression
  - Parametric vs. Non-Parametric
- OSL & why OSL not valid in actuarial application
- Insurance data distribution & variance structure
- R basic

GLM

- Three components: random(exponential family), systematic (linear predictor); link function
- Assess & diagnosis GLM model: deviance/log(likelihood), AIC/BIC, deviance residuals
- Offset & weights

Clustering

- General idea (what/why/how), different types
- K-means, # of cluster/how, “separation”, strength/weakness
- Hierarchical, basic, interpretation
- How to build a clustering model; and how to read & understand a cluster model in R

Decision Tree

- General idea; classification & regression
- How to split, & when to stop; greedy/prune
- Model interpretation; how to read model results